The Imagination Machine

APF BASIC

Language Reference Manual

apf electronics inc.

The Imagination Machine

APF BASIC

This version of the popular BASIC computer language has been specifically designed and written for the APF Imagination Machine. Special features of this language include:

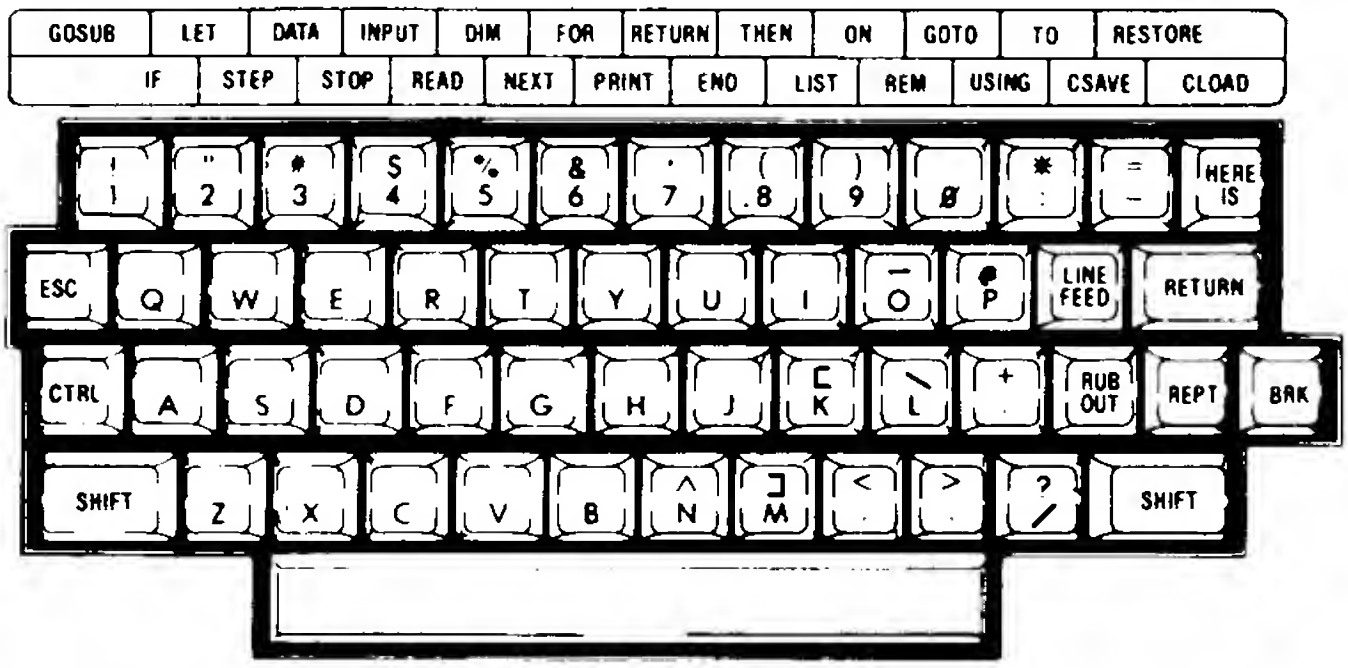
- Three octave music system.
- High and low resolution color graphics.
- Hand control input commands.
- Four place, fixed decimal arithmetic with 12 digit accuracy.

The information in this manual is organized and presented for use by experienced programmers. For an introduction to BASIC programming, we suggest BASIC Tutor™ —an interactive teaching system available for your APF personal computer.

INFO/INDEX

CHAPTER 1 KEYBOARD AND HAND CONTROLS

Study the keyboard carefully. Most of the keyboard is like a standard typewriter with several special keys. In addition to the special keys the Imagination Machine also has two rows of control keywords across the top of the keyboard. Use of the special keys is described in the section below.



ESC
When ESC is pressed a | appears on the screen.

CTRL
The CTRL key allows you to use the control keywords printed above the top row of keys. The two rows of control keywords correspond to the top two rows of keys. To print GOTO on the screen hold down the CTRL key and press the Ø key. To print CLOAD on the screen hold down the CTRL key and press the RETURN key.

SHIFT
Since the computer has no lower case letters the SHIFT keys are only used to print the symbols on the upper half of the number keys and wherever shown on other keys.

HERE IS
Is not used in the current version of APF BASIC.

LINE FEED
Is not used in the current version of APF BASIC.

RETURN
The RETURN key acts like the carriage return on a typewriter. It is also used to tell the computer when you are through entering. When this key is pressed the computer will look at what you have typed and decide what to do with it.

RUB OUT
The RUBOUT key is used to back space and erase one character at a time.

REPT
The REPT (repeat) key is used to make any other key print continuously on the screen as long as you hold down both keys, up to the maximum line length of 128 characters.

BRK
The BRK (break) key causes your program to stop running and return to user input or control mode. It does not function during an input statement or while saving and loading on tape.

HAND CONTROLS
There are two hand controllers attached to the MP 1000. One is marked with an L for left and the other with an R for right. They can be accessed with a key $ function (P 12).

VARIABLES

There are two types of variables, number variables or letter variables. The maximum number of variables is 26.

NUMBER VARIABLES—Number variables can be up to 13 digits long ranging from +999,999,999.9999 to −999,999,999.9999.

LETTER VARIABLES—Letter variables can contain up to 100 characters. The maximum length of each letter variable must be defined with a dimension statement.

VARIABLE NAMES—Variable names can be up to 5 characters long, but only the first 2 characters are used. Names cannot contain a keyword. The first character must be a letter. Letter variable names must end with a $.

EXAMPLE: Variable names that are correct but different.

A,A$, A1, A1$, A2SAM, A2SAM$

Variable names that the computer sees as being the same.

AL, ALSAM, ALLEN, ALL, AL123.

Variable names that are illegal and don't work.

3AL, A$B, A3#.

ALGEBRAIC OPERATORS

The symbols used for arithmetic operations are

+	a+b:	arithmetic addition:	a plus b
−	a−b:	arithmetic subtraction:	a minus b
*	a * b:	multiplication:	a times b
/	a / b:	division:	a divided by b
↑	a ↑ b:	exponetiation:	a to the b power—note: b must be an integer.

RELATIONAL AND LOGICAL OPERATORS

Used to compare 2 values, produces a true or false result.

a = b	a equal b
a<>b	a not equal b
a < b	a less than b
a > b	a greater than b
a<=b	a less than or equal b
a>=b	a greater than or equal b

RULES OF PRECEDENCE

Expressions are evaluated with operators having the following precedence

highest precedence: negation
↑ exponention
*,/ multiplication/division
−,+ addition/subtraction
lowest precedence: relational

When 2 operators of the same precedence occur, the expression is evaluated left to right.

LINE LENGTH

Maximum line length is 128 characters. Keywords count as one character.

SPACING

All spaces are ignored except those inside quotation marks or in REM statements, or in a print using definition.

MULTIPLE STATEMENTS/LINE

Multiple statements are separated by a colon.

ex: 10 PRINT 123: IF A=B THEN GOTO 30

MAX LINE

Maximum line #, is 9999. All line #'s must be integer only.

CHAPTER 3 COMMANDS

EDIT
The EDIT statement is used to change a portion of a line in your program. It must be followed by the line number you want to change and the ESC key. Then type the section of the statement you want to change. Press the ESC key again and then the new characters.

EXAMPLE: EDIT N ESC X ESC Y RETURN
N is line # to be edited
X is text to be deleted
Y is text to replace X

EXAMPLE: 100 INPUT A$:PRINT IN$
200 IF IN$="NOT" THEN 1000
300 IF IN$="YES" THEN 100
EDIT 100[A$[IN$
EDIT 200[T[
EDIT 300[100[RETURN
LIST
100 INPUT IN$:PRINTIN$
200 IF IN$="NO" THEN 1000
300 IF IN$="YES" THEN RETURN

LIST
Lists all lines in your program

LIST 100
Lists line 100 of program

LIST 100,
Lists from line 100 to end of program

LIST 100,5
Lists five lines of your program, beginning with line number 100

RUN
Sets all variables to zero. Execution then begins with the first line in your program.

CHAPTER 4 PROGRAM STATEMENTS

CALL
Transfers program execution to a machine language routine written in M 6800 object code

EXAMPLE: CALL 17046
This call branches your program to a special sub-routine that begins at location 17046. This particular sub-routine will clear the TV screen. When a 6800 RTS instruction occurs, return will be to the next basic statement.

DATA 5,10,BILL
Stores data to be accessed by READ statements. Data items will be read sequentially. Items may be numbers or characters but not expressions. Each data item must be separated by a comma

DIM
Reserves memory for strings, lists of strings, or tables of numbers.

EXAMPLE: DIM A$(10),B$(4,12),C(5,8)
This example reserves memory for an 11 character string, (A$), 5 strings of 13 characters each, (B$), and a table of numbers that is 6 rows by 9 columns (C) Maximum string length is 100 characters, A$(99).

END
Stops program and returns control to the console.

FOR A=B TO C
Combines with a NEXT statement to form a loop. Variable A begins with the value of B and increments to the value of C.

FOR A=10 TO 1 STEP-1
Combines with a NEXT statement to count from 10 to 1. Note: if step value is not given, it is assumed to be 1.
EXAMPLE: 10 FOR A = 10 to 1 STEP -1
20 PRINT A
30 NEXT A

GOSUB 100
Calls a BASIC subroutine located at line 100.

GOTO 100
Transfers program to line 100.

IF A=B THEN 300
Transfers program to line 300 if A equals B. If A is not equal to B the rest of the line is not read by the computer and the program transfers to the next numbered line. Any comparison can be used, such as;
= equal
<> not equal
> greater than
< less than
>= greater than or equal
<= less than or equal

IF A$<=B$ THEN GOSUB 2000
Transfers program to the subroutine at 2000 if A$ is less than B$. The value of A$ is compared to the value of B$ alphabetically. Numerals come before letters alphabetically.

IF A = B THEN PRINT "EQUAL":END
Prints the word EQUAL and ends the program if A and B are not equal. If A equals B then the next line of the program is executed.

INPUT A
Reads data from the keyboard. A question mark is displayed to indicate that the computer is waiting for a response. Separate multiple input items with a comma. Command is only terminated after a return key is pressed.
EXAMPLE: INPUT A,B

The user must respond with two numbers separate by commas. If only one input is given, then computer automatically asks for second one (B).

INPUT A$,B
The user must enter a character string comma and a number.

INPUT "AMOUNT", A
Prints the word AMOUNT followed by a question mark and waits for the user to type in a number.

NOTE: If a number is being requested and the user responds with a group of characters beginning with a letter, then the computer accepts the input as if he types a Ø.
If the user types a group of characters beginning with one or more numerals the computer accepts those numerals as the number.

LET A=5 or A=5
Sets the variable A equal to the constant 5. The word LET is optional and A=5 also works. The constant 5 can be replaced by an arithmatic expression.
EXAMPLE: A=3*(B+C)/4
B=3+B/C-A^2*C
This example is equivalent to B=3+(B/C)-((A^2)*C)

ON A GOSUB 1000,1200,1500,2000
Calls one of a list of subroutines, depending on the value of A. If A equals one, GOSUB 1000 results; if A equals two, GOSUB 1200 results; and if A equals three, GOSUB 1500 results. If A equals any other value, then GOSUB 2000 results.

ON A GOTO 10,20,30
Jumps to one of a list of lines, depending on the value of A. If A equals one, GOTO 10 results and if A equals two, GOTO 20 results. Any other value results in a GOTO 30.

POKE 784,1
Places the number 1 in memory location 784. This POKE displays the letter A in the middle of your screen. Any memory location can be poked as long as it is between 0 and 65, 536. The value poked must be a positive integer from 0 to 255. If greater than 255, modulo (255) is performed

PRINT A
Prints the value of A.

PRINT A,B
Prints the values of A and B in separate columns. Columns are automatically 8 characters wide. (ie.A is column 1, B is printed in column 9).

PRINT A;B
Prints the values of A and B with no spaces between them.

PRINT A;TAB(10);B
Prints the value of A, spaces ten characters from the left margin, and prints the value of B.

PRINT A;SPC(10);B
Prints the value of A, spaces ten characters from the last character in A, and prints the value of B.

PRINT "HELLO"
Prints HELLO.

PRINT USING "$$###.##",A
Prints the value of A with a leading dollar sign and two decimal places . . . $29.50, $104.01, $.02, etc. You must use as many pound signs (#) as there are digits in the value of A.

PRINT USING "######## TOTAL IS ###" A$,B
Replaces the pound signs (#) with the characters in A$ and the value of B . . . SATURDAY TOTAL IS 128. DOOMSDAY TOTAL IS 27. TOTAL IS 365, etc.

PRINT USING A$,B
Prints the value of B, using the characters in A$ as the format. The format string is contained in the string variable. A$

PRINT USING 100,A
Prints the value of A, using the data in line 100 as the format. Line 100 must begin with a colon and quotes are not required.

```
100 :##.###
100 :$$####.##
100 :LAST QUOTE: ####
```

READ A,B$
Reads data into specified variable from DATA statement.

```
EXAMPLE: 100 DATA 37,BILL,SAM,8
         200 READ A,B$
         300 READ C$,D,E,F$
         310 DATA 42,THE
```

Note that DATA statements can be placed after the READ statements.

RESTORE
Resets the data pointer so that the next READ statement begins with the first DATA statement in the program.

```
EXAMPLE: 100 DATA 37,BILL,8,SAM
         200 READ A,B$
         300 RESTORE
         400 READ C,D$
```

In this example C will be assigned the value 37.

RETURN
Returns from the subroutine to the statement following the GOSUB.

STOP
Stops the program.

CHAPTER 5 ARITHMETIC FUNCTIONS

All functions are written of the form
NAME(X)
Name is the function mnemonic. No spaces are allowed between the name and the left parenthesis.
X is the operand and can be a constant, expression or function.

ABS(X)—returns absolute value of x

```
ABS(−1.2) = 1.2
ABS(0) = 0
ABS(1.45) = 1.45
```

INT(X)—returns integer portion of X. Fractional portion of x is truncated.

```
INT(12.34) = 12
```

PEEK(X)—returns decimal value stored in memory location x. X is a decimal address.

EXAMPLE: A=PEEK (41452)
In this example the value of memory location 41452 is placed in A. If it is Ø then CSAVE will place the TV image at the beginning of your tape.

RND(X) random number between Ø and Ø.9999

SGN(X) 1 if x>Ø, −1 if x<Ø,Ø if x=Ø.

ASC(A$) Returns ASCII Code of the first character in A$.

```
ex: PRINT ASC("A")
    65                          This is decimal ASCII code for letter A.
```

CHR$(N) Produces string of one character length whose ASCII code is N.

```
Ex: PRINT CHR$(31)
    ←                           This is the literal whose ASCII code is decimal 31.
```

LEN(A$) Returns # of characters (Length) in A$.

```
EX: DIM A$(10)
    A$ = "ABC"
    PRINT LEN (A$)
    3                   Length is 3.
```

KEY$(0) Reads the typewriter keyboard. If no key is pressed, A$="". If key is pressed, A$="C" where C is the character typed.

```
EXAMPLE: 10 A$=KEY$(0):  IF A$="" THEN 10
         20 PRINT A$
```

Line 10 is asking if any key on the keyboard is pressed. If no, it loops back to the beginning of line 10 and asks again. If a key is pressed, the program drops through to line 20 and prints that key on the screen.

A$=KEY$(2)

Reads the left hand control. If no key is pressed, A$="". If key 0 to 9 is pressed, A$ "n" where n is the number of the key. If the knob is moved, A$="d" where d is the direction (N,S,E, or W). If CL is pressed, A$="?". If either EN or the FIRE button is pressed, A$="!".

A$=KEY$(1)

Reads the right hand control in the same manner as described above.

ADDRESSING CHARACTERS IN A STRING

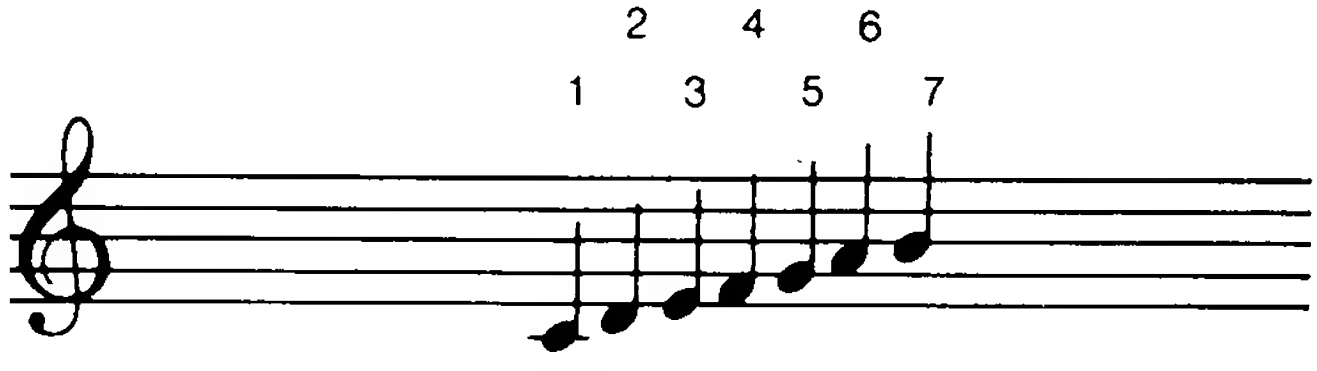
Since strings are dimensional as arrays, it is possible to address any character or group of characters in a string.

EXAMPLE:

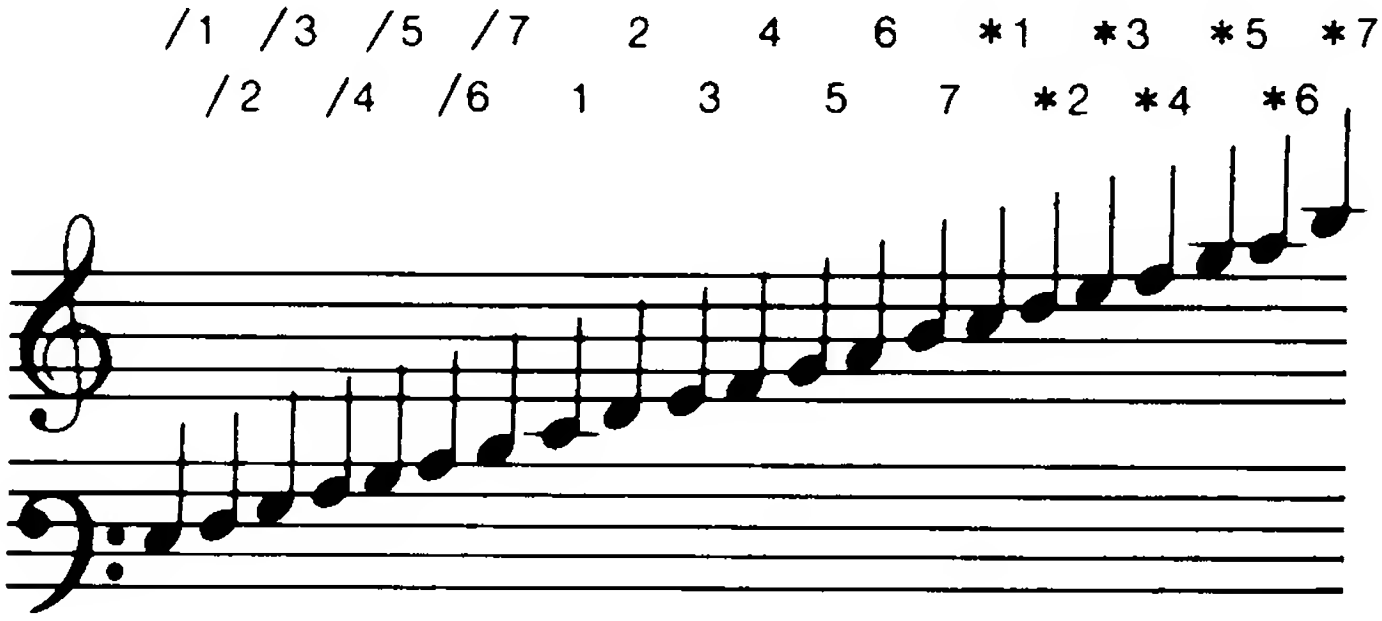
```
DIM A$(9),B$(1)
A$  "ABCDEFGHIJ"
PRINT A$                    This prints ABCDEFGHIJ
PRINT A$(4)                 This prints EFGHIJ
B$  A$(3):PRINT B$          This prints DE
```

CHAPTER 7 MUSIC

Using the MUSIC command, you can write and play music on your computer using numbers. The numbers 1 through 7 play the musical scale.



To play one octave higher, place a multiply sign (*) in front of the number, and to play one octave lower, place a divide (/) sign in front of the number.



You can also play sharps and flats by typing a + or − sign in front of the note.
The number zero (Ø) is used to hold the previous note.
Leaving a space between numbers creates a pause in the music.

EXAMPLE: MUSIC"3212333"
MUSIC"3212333000"
If the music plays too quickly you can slow it down by placing a Ø between each note.
MUSIC"302010203030"
Spaces create a different sound.
MUSIC"3 2 1 2 3 3 3 "
If you want to use the same sequence of notes over and over you can assign them to a variable.
MUSICA$,B$
EXAMPLE: 10 For I=1 to 10
20 Music "/1/2/3/4/5/6/71234567*1*2*3*4*5*6*7"
30 Next I
EXAMPLE: Let's try a song.
MUSIC "32123 3 3 2 2 2 3 5 5 32123 3 3 3 2 2 321"
EXAMPLE: Using A$,B$
A$="5552332"
B$="77665"
MUSIC A$, B$

CHAPTER 8 GRAPHICS (64x32)

The low resolution graphics mode divides the screen into 512 cursor size rectangles, 32 columns wide by 16 rows high (32 x 16). These are the same 512 boxes used to show alphanumerics and therefore this mode allows mixing of alphanumerics and color graphics on the screen.

COLOR

Any cursor position can be any one of eight colors at any time by assigning it a color number from Ø to 7. Setting COLOR=expression determines the color to be used in the next line or plot command. Expression can be a constant or expression. Modulo arithmetic is performed so the expression can be greater than >7.

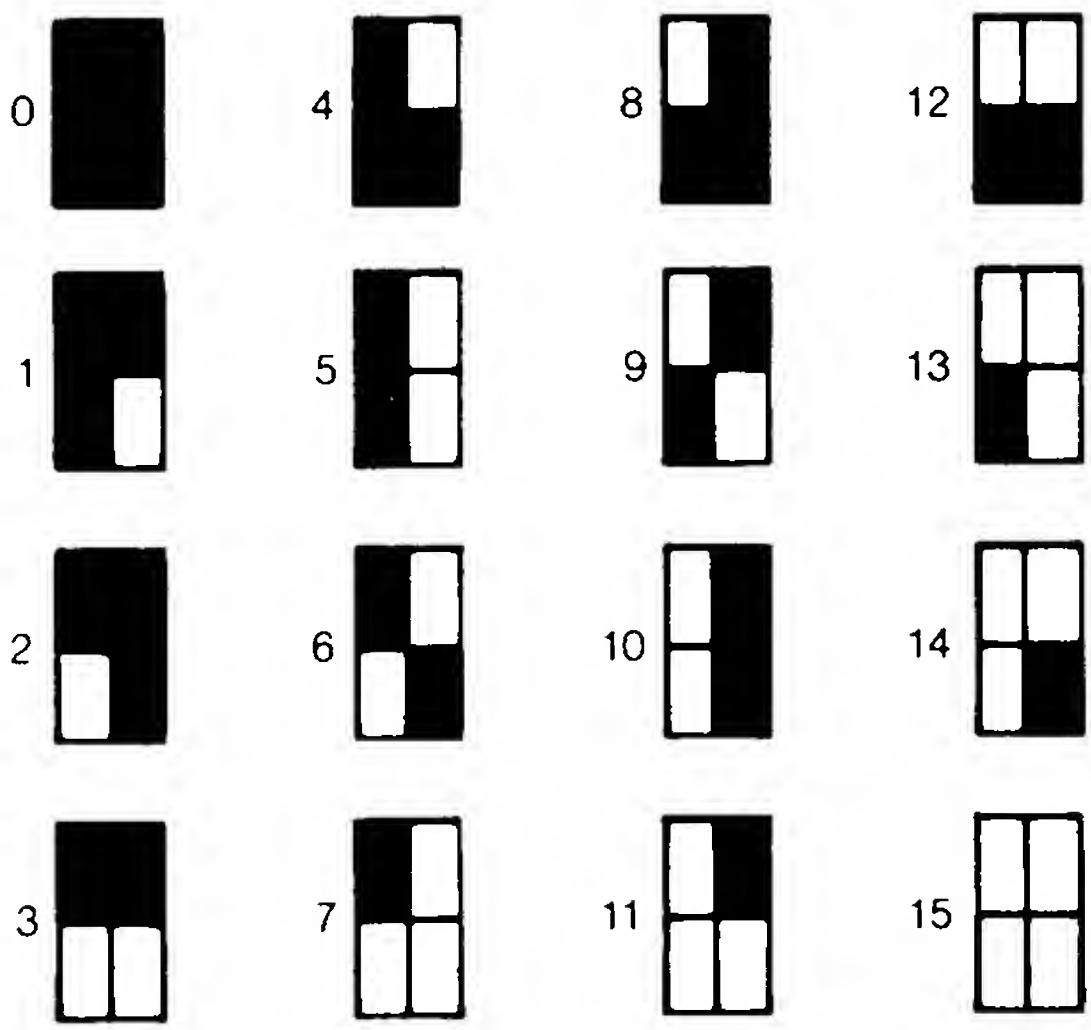
Ø = Dark Green	4 = White
1 = Yellow	5 = Light Green
2 = Blue	6 = Purple
3 = Red	7 = Orange

This means you can have eight different colors on the screen at the same time.

SHAPE

To get higher resolution each cursor position can be divided into four cells. By selecting the appropriate shapes you can draw a picture with much higher resolution (64 x 32).

The shape table shows the number for each cursor shape.



Shape is set equal to a constant or expression. It's value is used in the next plot or line command modulo 15 arithmatic is performed so shape can be greater than 15:

Since only one color can be assigned to a cursor position the cells that are lit will all be the same color and the remaining cells will be black.

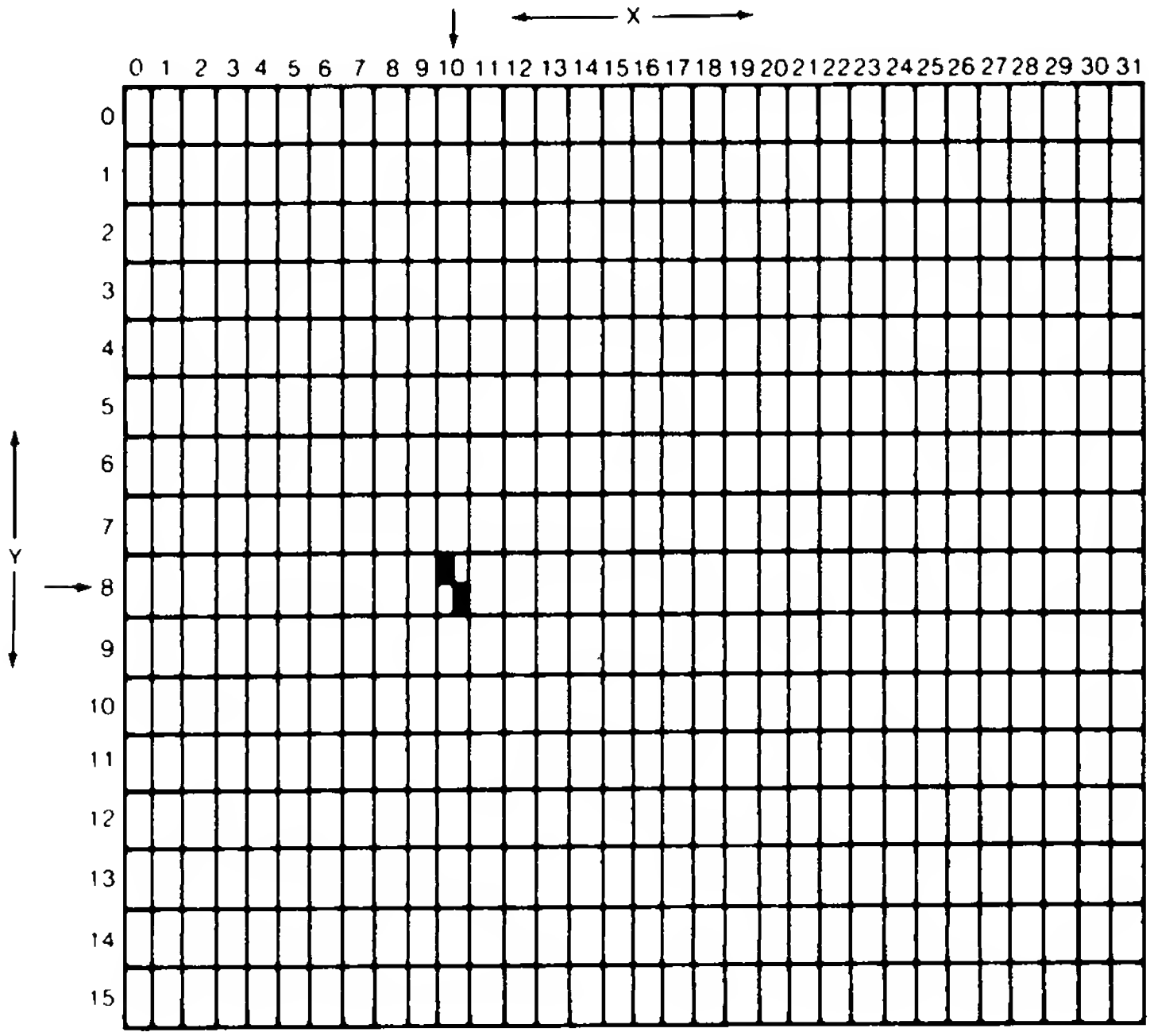
SHAPE 6 BLACK  YELLOW

COLOR 1 YELLOW BLACK

PLOT

To select the X and Y position on the screen where you want the shape to appear use the PLOT command.

PLOT 10,8
Means place the shape in the 10th column in the 8th row.



To place shape 6 on the screen in yellow and black in the position X = 10, Y = 8 the command would be:

```
COLOR=1:SHAPE=6:PLOT10,8
```

LINE
To draw a horizontal line of the selected shape use the HLIN command and specify the starting column, the ending column and the row.

HLIN 10,20,8
Means draw a line starting in column 10 ending in column 20 using row 8. The same process is used to draw vertical lines on the screen.

VLIN 8,12,10
Means draw a vertical line starting in row 8 ending in row 12 using column 10.

NOTE: If you plot a shape on top of another shape the second shape is added to the first shape; it does not replace it. The new shape takes on the new color. Plotting a 9 on top of a 6 gives you a 15.

EXAMPLE:

```
10 SHAPE=15
20 FOR I=0 to 7
30 COLOR=I
40 HLIN I,31-I, I
50 HLIN I,31-I, 15-I
60 VLIN I,15-I, I
70 VLIN I, 15-I,31-I
80 NEXT I
90 GOTO 90
```

CLEAR SCREEN
To clear the screen use CALL 17046

EXAMINE SCREEN LOCATION
A=PEEK(512+Y*32+X)
EXAMPLE:

```
PRINT PEEK(512+10*32+18)
```

This prints the contents of the 18th position of the 10th row The numbers 512 and 32 are always used as shown. The other two numbers determine the X and Y coordinate of the screen.

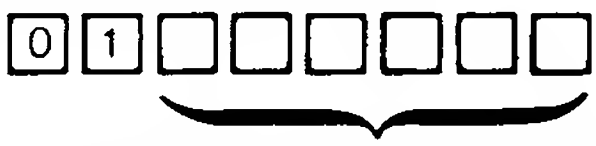
If A>127, the location contains a shape ( ,  , etc.)

If A< =127, the location contains a character (A, B, C, etc.)

0≤A≤63 light green character on dark green
A CHARACTER

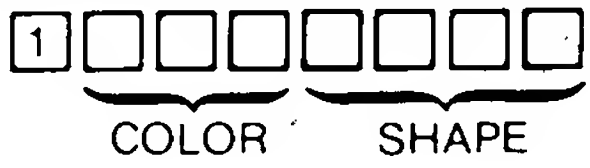
CHARACTER CODE (APPENDIX A)

64≤A≤127 dark green character on light green
A = CHARACTER + 64



CHARACTER CODE (APPENDIX A)

128≤A≤255 shape
A = 128 + COLOR*16 + SHAPE



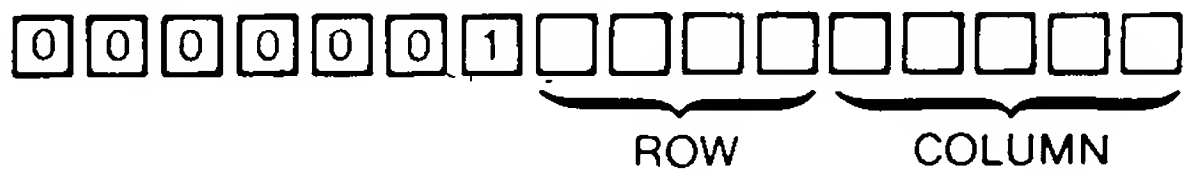
COLOR SHAPE

REVERSE VIDEO (BLACK/ORANGE)
POKE 8193,60
Changes color of reverse video characters from dark green on light green to black on orange. This does not affect letters in standard video, light green on dark green.

REVERSE VIDEO (GREEN/GREEN)
POKE 8193,52
Restores standard video. Use after POKE 8193,60.

LOCATE CURSOR
A Peek (40960)*256 + PEEK(40961)
Places in A the position of the cursor.

A ROW*32 + COLUMN + 512



ROW COLUMN

POSITION CURSOR
Position cursor by setting A as shown and poking two locations. To position cursor at 22,7

- L=7
- P=22
- A=L*32 + P + 512
- POKE 40960,A/256
- POKE 40961,A-INT(A/256)*256

CHAPTER 9 GRAPHICS (128x192)

Graphics mode (128 x 192)—There is a high resolution graphics mode which has the following features.

1. 128 x 192 resolution
2. 2 color sets each with 4 colors so there can be a total of 8 colors on the screen at once.

MODE SELECT

Enter this graphics mode with:

```
POKE 8193,60
POKE 8194,158
```

Return to the (64 x 32) graphic mode with:

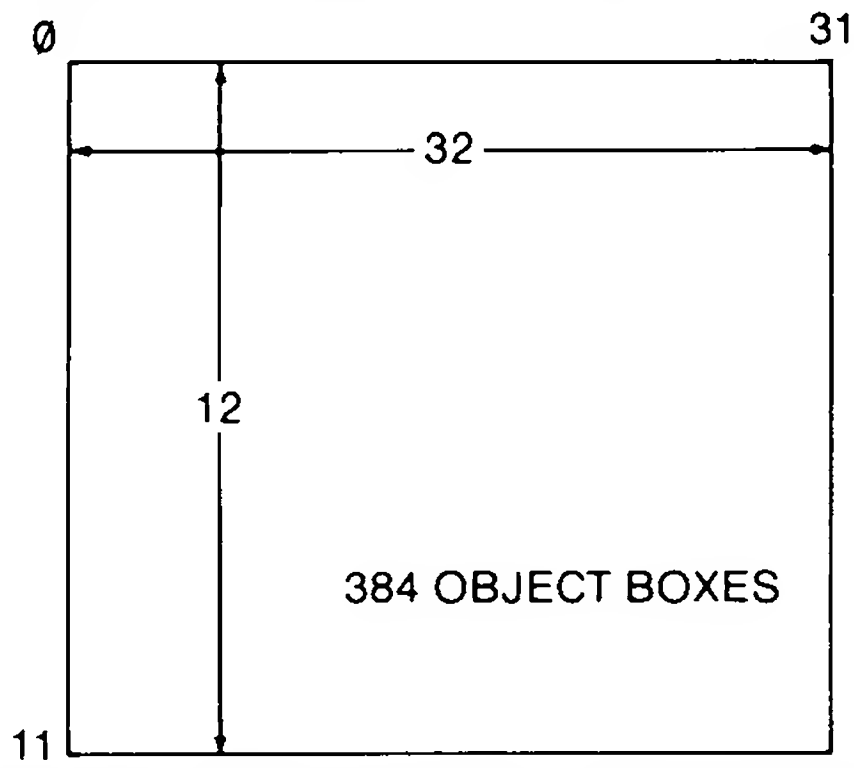
```
POKE 8193,52
POKE 8194,30
```

When the computer is in this mode there is no alphanumerics capability. Therefore, pressing keys or the keyboard will not produce letters on the screen but will cause the computer to take a directed action.

MEMORY MAPS

Two memory areas are used. The screen map occupies memory locations Ø to 383 and the shape table occupies locations 512 to 1023.

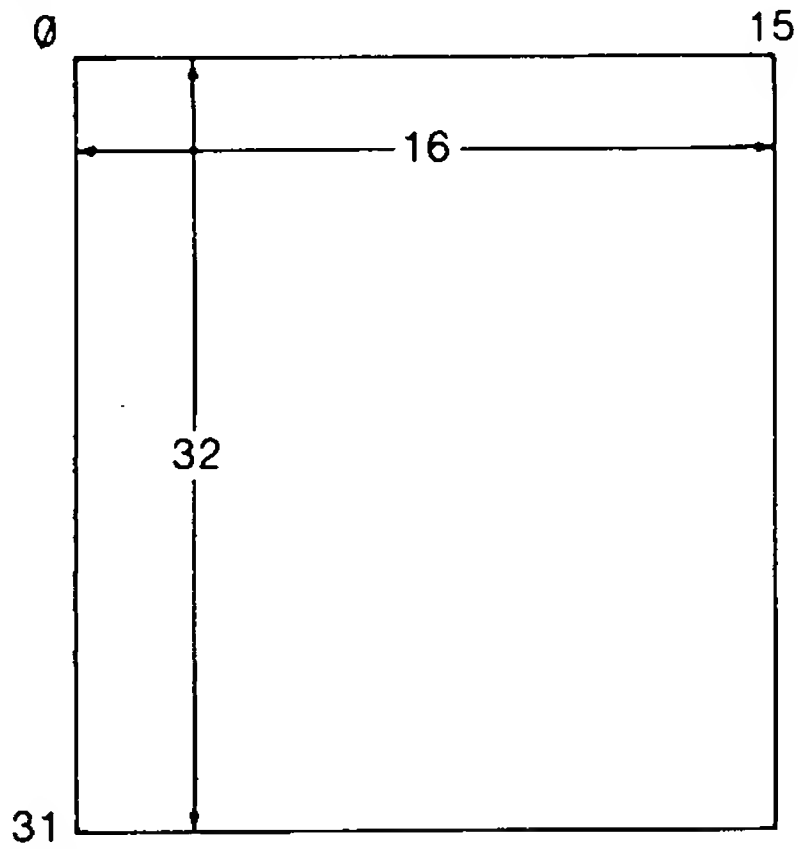
SCREEN MAP



The TV screen is divided into 384 object boxes (12 rows by 32 columns). Each box may contain any one of 32 special shapes.

These shapes are stored in the shape table and numbered Ø to 31.

SHAPE TABLE



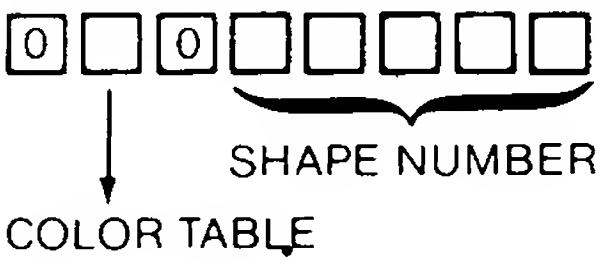
The shape table describes each of the 32 shapes that can be displayed. Each shape is composed of 16 bytes, one byte for each horizontal row of dots in the shape.

DEFINE SHAPES
In this graphic's mode each shape to be plotted on the screen is 4 dots wide and 16 dots high.

Each horizontal row of four dots is set with a POKE instruction. For example, this POKE sets the top row of dots in shape number one to the colors C1, C2, C3, and C4:
POKE 512,C1*64+C2*16+C3*4+C4
Each color number is between Ø and 3, depending on the color selected. The color tables are in the next section.

The example at the end of this section shows how up to 32 shapes can be created with similar POKE instructions.

PLOT SHAPES
After the shapes have been created in the shape table, they can be plotted on the screen.
The format for each byte in the screen map is:



If color table select Ø
Ø GREEN
1 YELLOW
2 BLUE
3 RED

If color table select 1
Ø WHITE
1 AQUA
2 PURPLE
3 ORANGE

NOTE: The color table is selected in the screen map and the specific color in the table is included in the shape definition.

EXAMPLE:

```
100 L=6:REM LINES ARE Ø TO 11
110 P=19:REM POSITIONS ARE Ø TO 31
120 S=13:REM SHAPES ARE Ø TO 31
130 T=Ø:REM COLOR TABLES ARE Ø OR 1
140 DATA Ø,Ø,1,1:REM A SHAPE IS 4 DOTS ACROSS
150 DATA Ø,Ø,1,1:REM BY 16 DOTS DOWN
160 DATA Ø,1,1,2:REM
170 DATA Ø,1,1,2:REM EACH DOT IS ONE OF FOUR
180 DATA 1,1,2,2:REM COLORS
190 DATA 1,1,2,2:REM
200 DATA 1,2,2,3:REM EACH COLOR IS NUMBERED Ø TO 3
210 DATA 1,2,2,3
220 DATA 2,2,3,3
230 DATA 2,2,3,3
240 DATA 2,3,3,Ø
250 DATA 2,3,3,Ø
260 DATA 3,3,Ø,Ø
270 DATA 3,3,Ø,Ø
280 DATA 3,Ø,Ø,1
290 DATA 3,Ø,Ø,1
300 REM
310 REM SET UP THE SHAPE
320 REM
330 FOR I Ø TO 15
```

```
340 READ A,B,C,D
350 POKE 512 +S*16+I,A*64+B*16+C*4+D
360 NEXT I
370 REM
380 REM PLACE THE SHAPE ON THE SCREEN USING
390 REM THE CHOSEN COLOR TABLE
400 REM
410 POKE L*32+P,S+T*64
```

CHAPTER 10 TAPE

Tape Operation
The tape system is specially designed for the Imagination Machine. The key features are:

1. Dual track tapes are used. One for standard audio recording or playback, and the other for digital recording or playback.
2. Microphone jack built in for audio recording. Any standard cassette microphone may be used.
3. Separate speaker built in for audio playback. Volume control is included.
4. Semi-automatic computer control. Record or playback of either audio or digital is under computer control. Fast forward and rewind are manually operated.
5. High speed digital transfer rate—approximately 1500 bits per second.
6. Tape counter included to allow manual location (thru fast forward or rewind) of a desired section of tape.

SAVING PROGRAMS
CSAVE is used to save program memory. It is not necessary to press the record button on the tape deck since that is used only to record audio. Do not delay more than 5 seconds when pressing the play/save button and the return key. For an 8K system CSAVE takes abou 45 seconds.
During a CSAVE all keyboard functions are inoperative: CSAVE will save the entire amount of random access memory.

Saving 512 Block for Screen
In addition to the program memory, CSAVE first saves a block of 512 bytes. This will be placed on the screen first when loading so pictures or messages can be stored. There are 2 possible blocks that can be saved. The selected one is determined by the contents of memory location 41452.
If (41452) = 0 then locations 512-1023 (screen memory) are saved.
If (41452) = 255 then locations 0-511 (scratch memory) are saved.

Saving Program Data
Since CSAVE always saves all memory, then all dimensioned variables are also saved on tape. This can be used to save and revise data. Simply use a goto statement to start the program after loading (run automatically clears all variables).
Note: After a reset is pressed, one RUN command is always necessary to initialize certain system constants. Therefore we recommend that if you want to use a tape with data, do the following sequence.

```
1 CLOAD
RUN                                   1st RUN after reset
Follow instructions to load
When OK appears
Type GOTO x where x is first statement of loaded program.
```

LOADING TAPES
CLOAD is the command for loading tapes. The first 512 bytes of the tape are always put to screen memory. After that the next 8K bytes are read to program memory. At the end of loading there is a check sum to indicate all data is read correctly. If it was, the word OK appears on the left side of the screen and the cursor returns. If anything except OK appears, it was a bad load and the tape must be reloaded.

AUDIO RECORDING

One of the nicer features of the Imagination Machine is that you can record your voice, or even music, on the same tape as your computer program. To do this, plug a microphone into the microphone jack on the body of the keyboard unit and press the record key. Now type in the following statement:

```
POKE 24578,62
```

This turns on the motor. Now talk or play music into the microphone. It is being recorded on the audio track of the tape without damaging or changing any programs you may have placed on the program track.

To turn off the motor, type:

```
POKE 24578,38
```

AUDIO SHUTOFF

To stop the voice track on a program tape from playing through the speaker, type:

```
POKE 24578,54
```

MOTOR START/STOP

To start the cassette tape motor and listen to the audio track, type:

```
POKE 24578,62
```

To stop the cassette tape motor, type:

```
POKE 24578,38
```

APPENDIX A

RESERVED WORDS

The following words can not be used as or contained variable names.

ABS
ASC
CALL
CHR$
CLOAD
COLOR
CSAVE
DATA
DIM
END
FOR
GOSUB
GOTO
HLIN
IF
INPUT
INT
KEY$
LEN
LET
LIST
MUSIC
NEXT
ON
OPEN
PEEK
PLOT
POKE
PRINT
READ
RESTORE
RETURN
RND
RUN
SGN
SHAPE
SPC
STEP
STOP
TAB
THEN
TO
USING
VLIN

APPENDIX B

ASCII CODES AND TOKENS

The following are codes used to place alphanumerics on the screen:

Decimal	Will Appear	Decimal	Will Appear
0	@	32	Space
1	A	33	!
2	B	34	"
3	C	35	#
4	D	36	$
5	E	37	%
6	F	38	&
7	G	39	
8	H	40	(
9	I	41	)
10	J	42	*
11	K	43	+
12	L	44	,
13	M	45	-
14	N	46	.
15	O	47	/
16	P	48	0
17	Q	49	1
18	R	50	2
19	S	51	3
20	T	52	4
21	U	53	5
22	V	54	6
23	W	55	7
24	X	56	8
25	Y	57	9
26	Z	58	:
27	[	59	;
28	\	60	<
29	]	61	=
30	↑	62	>
31	←	63	?

APPENDIX C

MACHINE LANGUAGE REFERENCE

The Imagination Machine contains a machine language monitor. You can use the monitor to create, display, change, and execute machine language programs. To use this appendix, you must be able to write programs in 6800 machine language. You must also have a working knowledge of hexadecimal notation.

CALL 28672

This BASIC statement takes you out of BASIC. You are now talking to the Imagination Machine Monitor. The Monitor puts a "*" at the beginning of each line on the screen. When you see the "*", you can enter one of the three monitor commands:

D nnnn where nnnn is a hexadecimal address
G nnnn where nnnn is a hexadecimal address
M nnnn where nnnn is a hexadecimal address

D nnnn – DISPLAY MEMORY

This command will display the 16 bytes of memory beginning at address nnnn. To display the next 16 bytes, press the "/" key. To end the command, press the RETURN key.

Example:
```
* D 9B3C
* 9B3C 20 E0 B6 A0 58 BD 9A B6 CE
  A0 9C 7E 9A 28 7C A0/
* 9B4C AA 20 D4 86 04 CE A0 AA 0C
  69 00 09 8C 80 9C 26 (Return)
```

G nnnn – GOTO MEMORY ADDRESS

This command acts much like the BASIC GOTO statement except the value NNNN is a four digit hexadecimal memory address. The computer immediately begins executing the machine language program at that address.

*G 8894

Address 8894 is the start of the Imagination Machine's BASIC. This is how you reenter BASIC. If you had a BASIC program in memory when you called the monitor, it should still be there.

M nnnn – MODIFY MEMORY

This command immediately displays the contents at memory address nnnn. You can do one of four things:

reply with the "/" key and the command proceeds by displaying the next position in memory.

reply with "↑" key and the command proceeds to display the previous memory position.

reply with the RETURN key and the command is ended.

reply with a two-digit hexadecimal number and the RETURN key and the command stores this new number in the current memory position. Then you can press Return, or ↑ with the results as above.

If the M command cannot change the memory location, it will respond with a "? ".

APPENDIX D

ERROR MESSAGES

The following is a list of error messages. If they occur during a program statement, the statement number as well as the message is displayed.

ARITHMETIC OVERFLOW

The result of a computation is greater than 999999999.9999. This problem can be eliminated by using the IF statement to test and limit the sizes of the variables in the equation before doing the computation.

DIMENSION

Something is wrong in Dimension Statement (size is zero or greater than 99, etc.).

DIVISION BY ZERO

The result of dividing by zero is undefined. The computer cannot proceed. This problem can be prevented by checking the value of a diviser with an IF statement before doing the division.

EXPRESSION

The expression is not properly formed. Examples are PRINT PEEK 123 – parenthesis not use around expression of a function. PRINT 234+3) – missing an open parenthesis. PRINT 1 ←2 – improper operator symbol in an expression.

EXPRESSION MISSING

An expression is expected to be located in a statement but is not found.

Example:
```
A = (Return Key)
For H =
```

EXPRESSION TOO LONG
You did nothing wrong. All computer languages have limits, and you just ran up against one of ours. This will occur if an expression has more than 8 nested brackets or parenthesis.

FOR – NEXT
A NEXT statement was executed when no FOR statement had been executed. Check your program. You have forgotten to supply the FOR statement or the logic of your program is incorrect. Somehow you branched to a NEXT statement without first going through its FOR statement. Fix the logic of your program before continuing.
Example: 10 next I
RUN

IF – THEN
The comparison in the IF statement is wrong. For example, IF A*B THEN 100. Fix the statement before running the program.

ILLEGAL MASK SIZE
The format defined for a PRINT USING statement is incorrect. For example, there may be more than 9 " # "s to the left of the decimal place, or more than 4 to the right. Correct the PRINT USING format, called a mask, before running the program.

ILLEGAL VARIABLE
This is a catchall error message when the interpretor finds something wrong with a variable. Some examples are:
A variable name starts with an illegal symbol – i.e., a number (1A=3). You have used more than 26 variable names. You used a dimensioned variable name that was not dimensioned.

MEMORY FULL
You ran out of memory. This will usually occur in a DIM Statement. If so, then reduce the allocation in a DIM statement or shorten your program.
Memory storage is as follows:
1. A numeric variable takes 7 Bytes. A dimensioned numeric array takes 7 Bytes for each element. Ex – DIM H (365) takes 7*365 Bytes.
2. Each line takes 2 Bytes for a line # plus 1 Byte for an end of line symbol.
3. All keywords (ie – PRINT, FOR, etc.) take 1 Byte.
4. Actual allowable user storage is 7166 Bytes.

A second reason for Memory Full can be due to your program's continuous execution of a FOR statement with no NEXT statement, or continuous executing a GOSUB with a RETURN.

NO – GOSUB
A RETURN statement was executed without a GOSUB having been executed. You have a logic error in your program. Make sure you do not use a GOTO or fall into a Subroutine. Fix the problem before running the program.

NO LINE # – REFERENCED
You are trying to GOTO a line number which doesn't exist in your program. Correct the statement before running your program again.

PRINT DELIMETER
There is an error in your PRINT statement. Check to see that each of the items being printed are separated by a comma or semi-colon. Correct the statement before running your program.

QUOTE MISSING
The right quote in a String Constant is missing. Correct the statement before running the program again.

READ – DATA
Either you have executed a READ statement when there is no DATA statement, or there is an error in the DATA statement. If there is an error in the DATA statement, correct it before running the program again.
If you are executing a READ statement, then either you forgot to include a DATA statement or you have tried to READ more variables than you have data in DATA statements. Make sure you have supplied all of the data in the DATA statements, or that you haven't executed a READ statement more times than you had intended.

WHAT
This is a catch-all. It usually means that one of the items in the line should have been a keyword but wasn't recognized as such. Carefully check the line and correct all errors before running the program again.

 # > 999999999.9999
You have a constant which has more than four decimal places or which is greater than 999999999.9999 or less than –999999999.9999. You will have to change your program so that numbers outside of this range are unnecessary.

APF electronics inc.
444 Madison Avenue New York, NY 10022

Printed in Hong Kong